

Coordinate Transformations

This tutorial describes how to perform coordinate transformations. In developing ioapiTools, I wanted to easily be able to use multiple types of coordinate systems. The function coordConv is the main transformation engine. It facilitates the translation between col row, xlon ylat (native coordinates) and lon lat (in degrees).

1. IOcoordConv

The iovar method for coordinate transformations is *IOcoordConv*:

```
## Convert single point at projection center
## lon/lat --> native coordinates
o3.IOcoordConv([(-90, 40)])
##
[(0.0, 0.0, 0.0)]
##

## Define a set of 3 points in xlon,ylat
coordIn = [(-66000, -102000), (66000, -102000), (66000, 102000)]

## native coordinates --> lon/lat
coordOut1 = o3.IOcoordConv(coordIn, iot.proj2llFlag)
print coordOut1
##
[(-90.784387646734501, 39.051042824601609, 0.0),
 (-89.215612353265513, 39.051042824601609, 0.0),
 (-89.191117262081093, 40.944189310030183, 0.0)]
##

## native coordinates --> col/row
coordOut2 = o3.IOcoordConv(coordIn, iot.proj2crFlag)
print coordOut2
##
[(17, 23, 0), (28, 23, 0), (28, 40, 0)]
##

## lon/lat --> col/row
coordOut3 = o3.IOcoordConv(coordOut1, iot.ll2crFlag)
print coordOut3
##
[(17, 23, 0), (28, 23, 0), (28, 40, 0)]
##

## col/row --> native coordinates at center of cell
coordOut4 = o3.IOcoordConv(coordOut2, iot.cr2projFlag)
print coordOut4
##
[(-66000.0, -102000.0), (66000.0, -102000.0),
 (66000.0, 102000.0)]
##

## col/row --> native coordinates at NW corner of cell
coordOut5 = o3.IOcoordConv(coordOut2, iot.cr2projFlag, iot.NWFlag)
print coordOut5
##
```

```

[(-72000.0, -96000.0), (60000.0, -96000.0),
(60000.0, 108000.0)]
##

## lon/lat --> native coordinates
coordOut6 = o3.IOcoordConv(coordOut1, iot.ll2projFlag)
print coordOut6
## 
[(-65999.9999999968, -102000.000000227, 0.0),
(65999.9999999968, -102000.000000227, 0.0),
(66000.00000000655, 101999.9999998313, 0.0)]
##

```

A couple of notes: First, the third element is the z coordinate, which is simply passed through to the underlying projection software and often is returned without modification. Second, transformations to column row simply determine which cell the point lies within. Third, the transformations from column row assume cell center, unless a specific corner is denoted (as in coordOut5). Finally, in coordOut6 the values are slightly different than the expected values of coordIn. This slight difference is due to round off error.

.Contents

2. coordConv

Instead of using the method, one can use the function call:

```

## get projection and domain info
ioM = o3.ioM.copy()
cdmsM = iot.cdmsmeta(o3, iot.cdmsvarFlag)

## col/row --> native coordinates at NW corner of cell
## Same transformation as coordOut5
iot.coordConv(ioM, coordOut2, iot.cr2projFlag, cdmsM, iot.NWFlag)
##
[(-72000.0, -96000.0), (60000.0, -96000.0),
(60000.0, 108000.0)]
##

```

The main difference between IOcoordConv and coordConv is that in the latter you need to pass the projection information (ioM) and the domain information (cdmsM) explicitly.

[Contents](#) [Previous](#) [Next](#)